Tema 3: Jerarquía de Memorias

• Memoria Principal

Departament d'Arquitectura de Computadors

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Catalunya





Índice

- Conceptos Básicos Memoria Cache
- Memoria Virtual
- Conceptos Avanzados Memoria Cache
- Memoria Principal
  - Introducción
  - Memorias de Semiconductores
  - Estructura interna de una DRAM
  - Evaluación y Optimización
- Conceptos Avanzados Memoria Principal

Introducción

Modelo de Memoria Principal

- **La Memoria Principal (MP) puede verse como:**

```
byte M[MemSize];
```

- **Operaciones básicas:**
  - Lectura: `dato = M[direccion];`
  - Escritura: `M[direccion] = dato;`
- **¡Atención! Desde el punto de vista del programador:**
  - La MP se direcciona a nivel de byte.
  - Los accesos a memoria pueden ser de múltiples tamaños: 1, 2, 4 u 8 bytes.
  - Si leemos 4 bytes en la dirección X, accedemos a las direcciones X, X+1, X+2 y X+3
  - Little endian vs big endian



Introducción

Tipos de Memorias

- **En función de la perdurabilidad:**
  - Volátil
  - No Volátil
- **En función del tipo de acceso:**
  - Sólo lectura (ROM, Read Only Memory)
  - Lectura / Escritura (RAM, Random Access Memory)
- **En función del tipo de uso:**
  - Primaria (semiconductores)
  - Secundaria (dispositivos de almacenamiento E/S, magnéticos y ópticos).
- **En función de la forma de acceso:**
  - Memorias de Acceso Secuencial (cinta VHS)
  - Memorias de Acceso Directo (DVD)

Memorias de Semiconductores

Tipos de Memoria de Semiconductores:

- **Memoria Estática** (SRAM, Static RAM). Cada celda de memoria equivale a 1 biestable (6-8 transistores). En comparación con las DRAM son **rápidas**, tienen un **alto consumo**, **pequeñas** (poca capacidad) y **caras**.

  → **Memoria Cache**

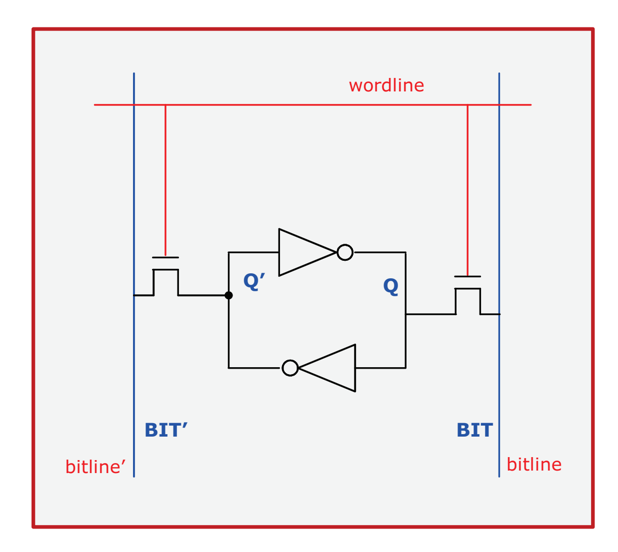
- **Memoria Dinámica** (DRAM, Dynamic RAM). Cada celda se comporta como un condensador (1-1.x transistores). En comparación con las SRAM son **lentas**, tienen un **bajo consumo**, **grandes** (mucha capacidad) y **baratas**. Problema del refresco.

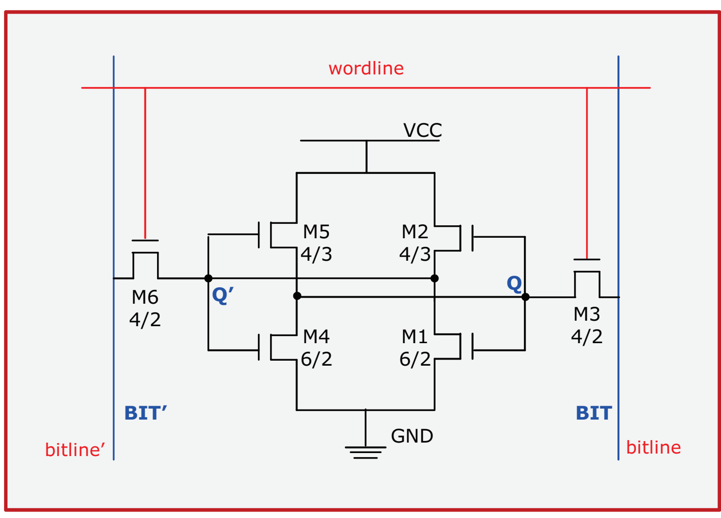
  → **Memoria Principal**



Memorias de Semiconductores

Celda SRAM de 6 Transistores

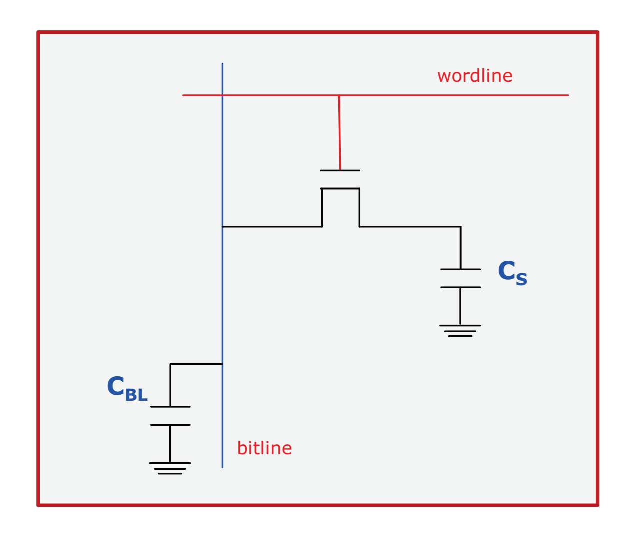


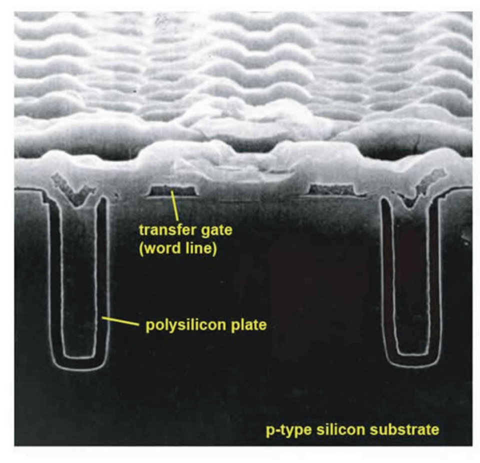


- La información se almacena en 2 inversores acoplados.
- Al activar la word line el dato almacenado se lee a través de las bit lines.
- Se obtiene el dato negado y sin negar.

Memorias de Semiconductores

Celda DRAM de 1 Transistor



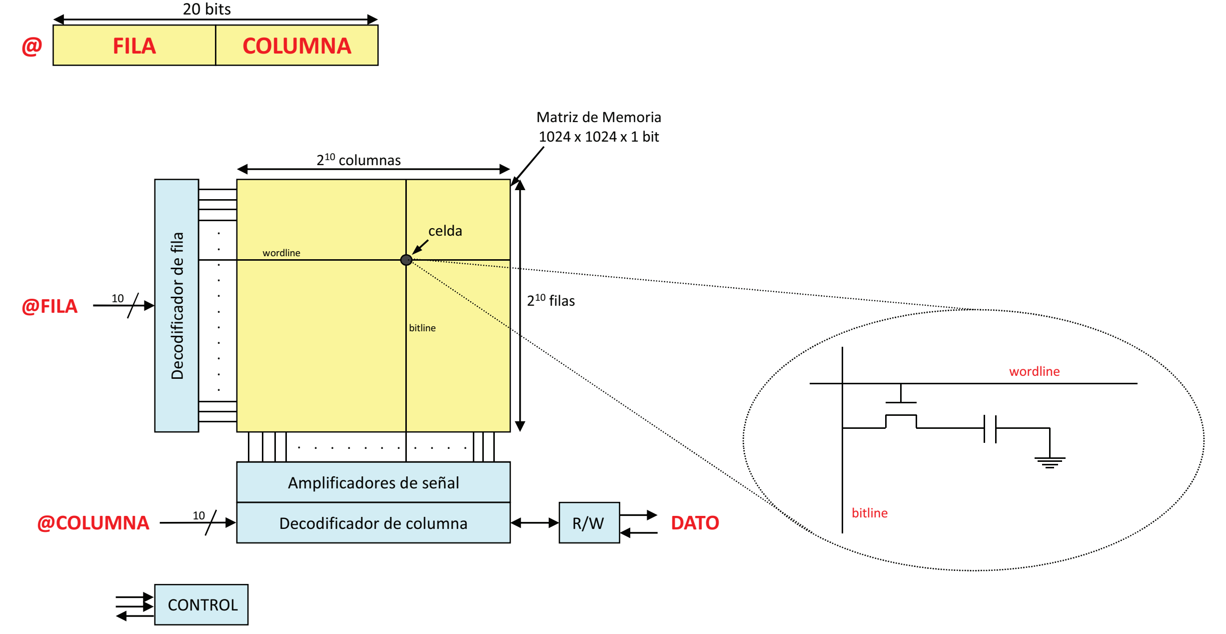


IEEE Solid-State Circuits Society

- La información se almacena en el condensador C_S.
- Al activar la wordline el dato almacenado en C_S se lee a través de la bitline.
- El condensador se va descargando poco a poco, es necesario recargarlo regularmente (refresco).

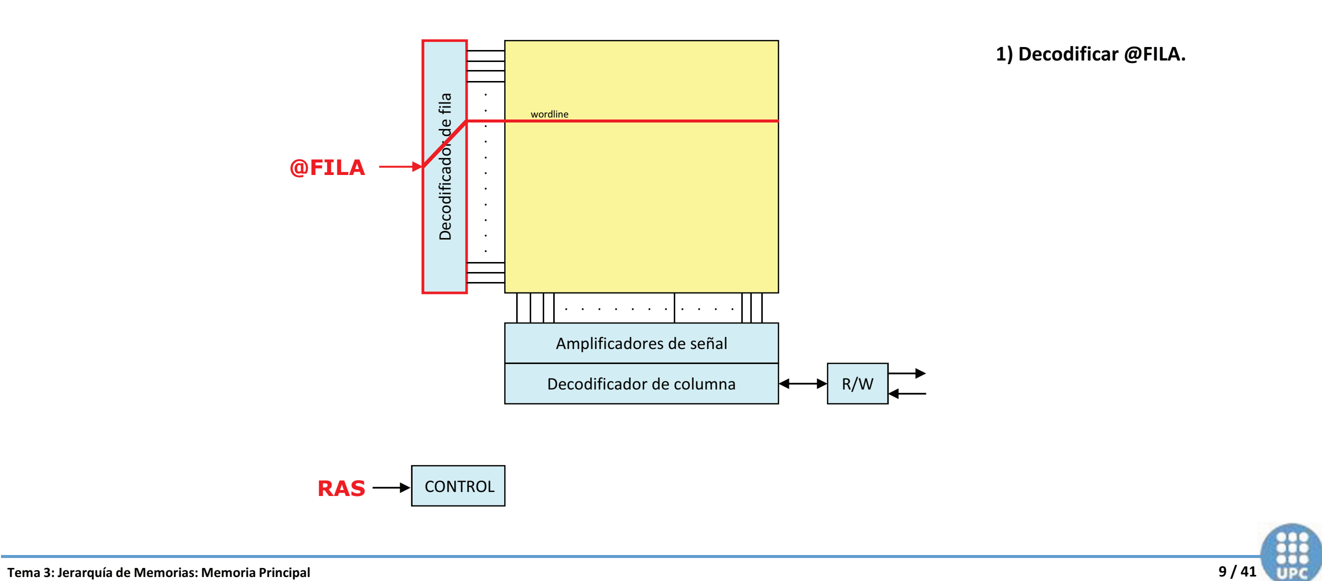


Estructura interna de una DRAM



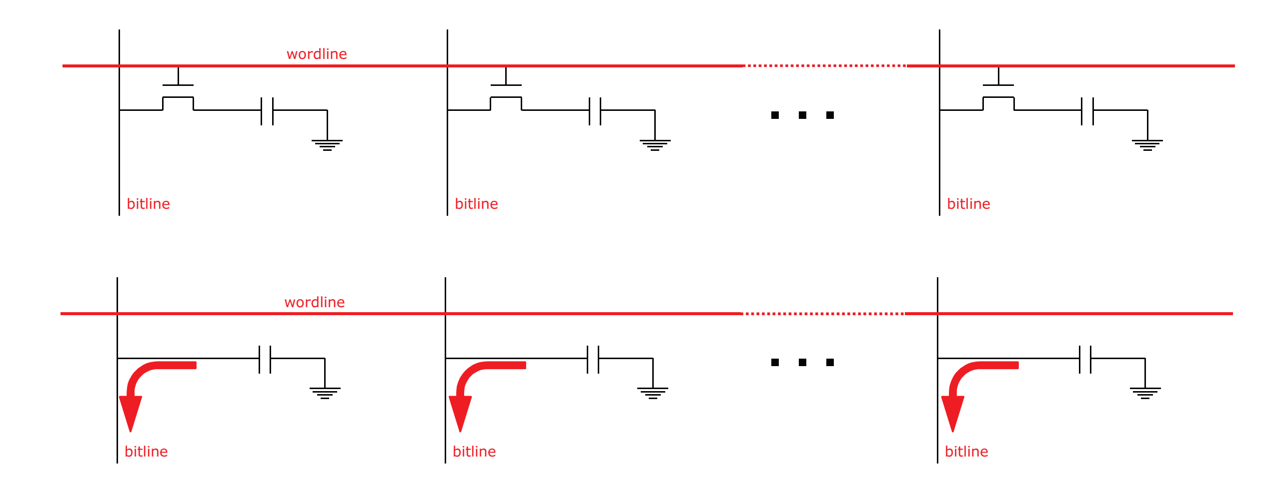
Estructura interna de una DRAM

- **Una operación de lectura:**
  1) Decodificar @FILA, se activa la señal wordline



Estructura interna de una DRAM

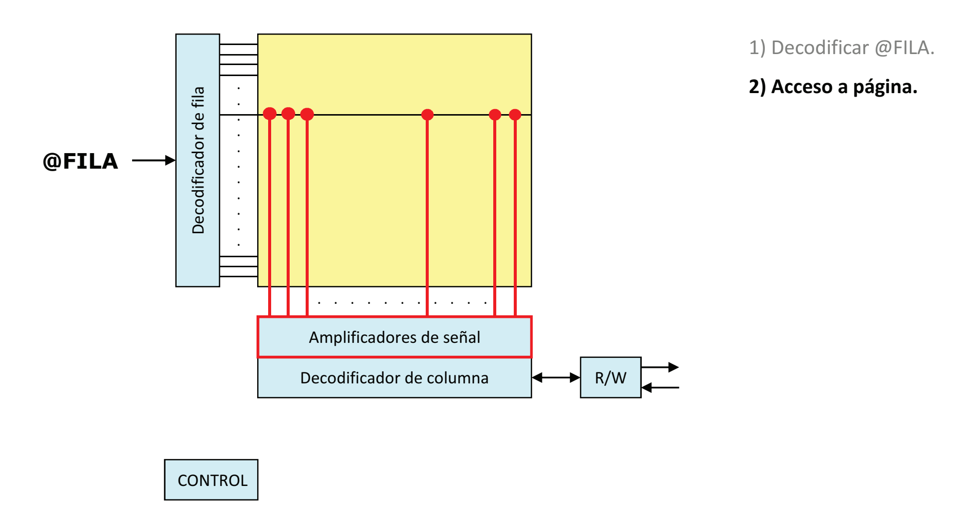
- **Una operación de lectura:**
  2) Se accede a todas las celdas de la fila, los datos de toda la fila se envían a los amplificadores de señal y se recupera la tensión (el dato está en un condensador que se va descargando poco a poco).



Estructura interna de una DRAM

- **Una operación de lectura:**

  2) Se accede a todas las celdas de la fila, los datos de toda la fila se envían a los amplificadores de señal y se recupera la tensión.

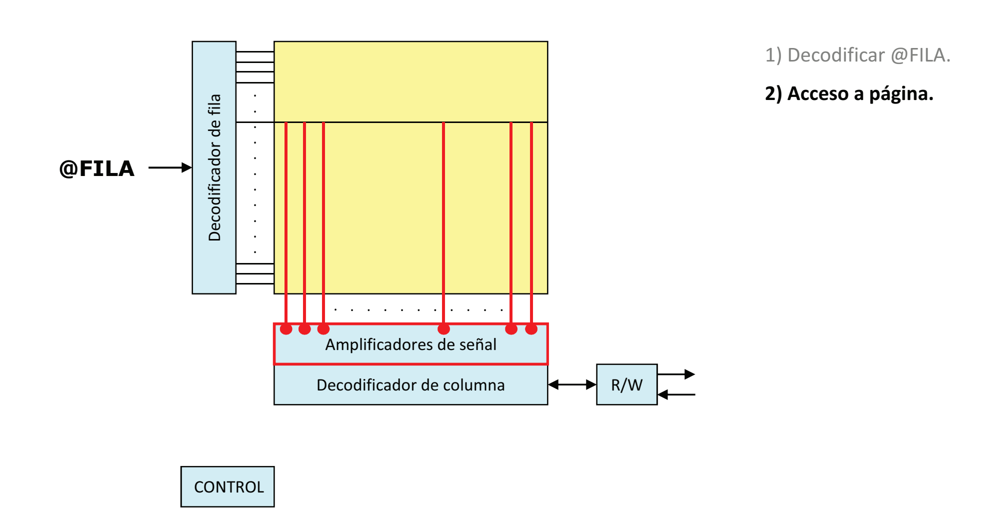




Estructura interna de una DRAM

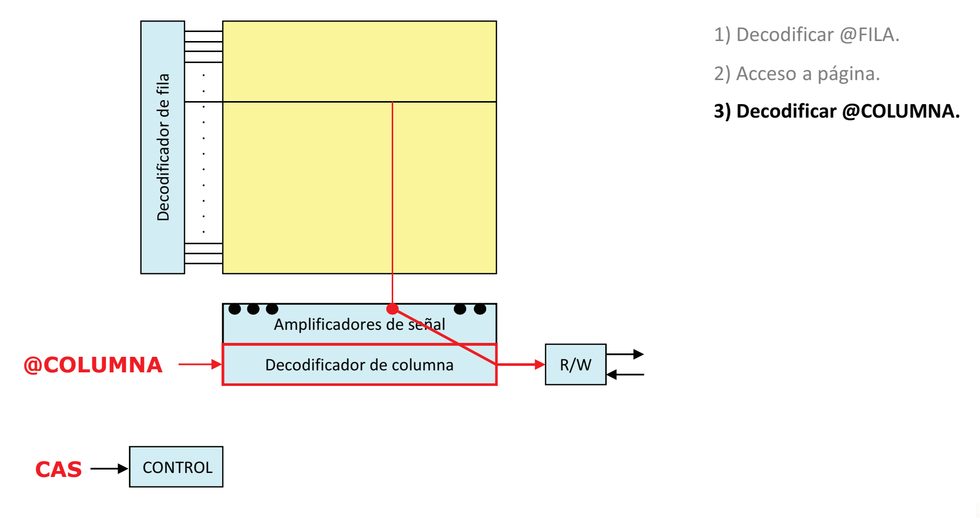
- **Una operación de lectura:**

  2) Se accede a todas las celdas de la fila, los datos de toda la fila se envían a los amplificadores de señal y se recupera la tensión.



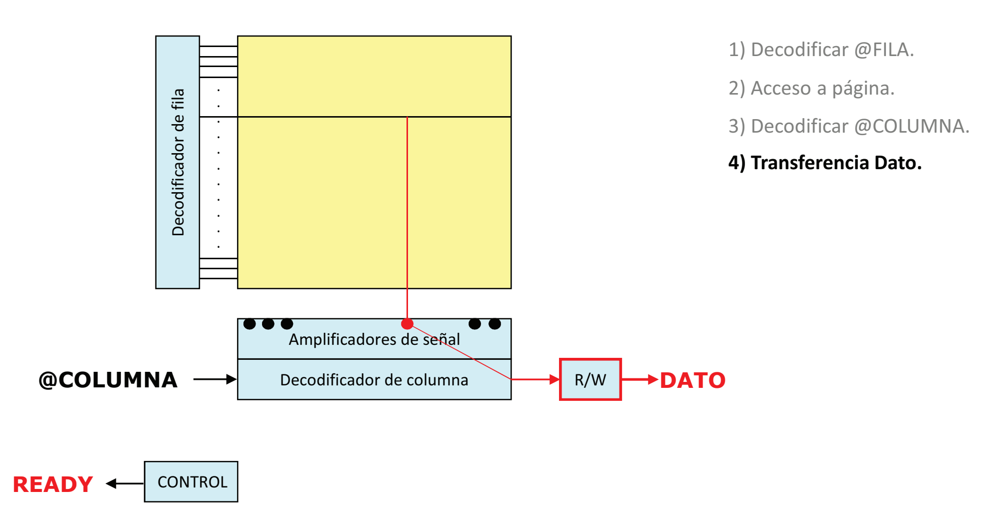
Estructura interna de una DRAM

- **Una operación de lectura:**
  3) Decodificar @COLUMNA, se selecciona una bitline y se envía el dato al buffer R/W.



Estructura interna de una DRAM

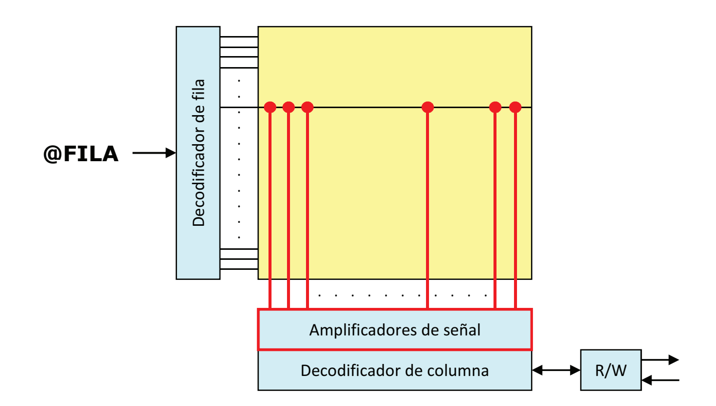
- **Una operación de lectura:**
  4) Se envía el dato al exterior desde el buffer R/W.



Estructura interna de una DRAM

- **Una operación de lectura:**

  5) La lectura es destructiva, hay que reescribir la celda (y toda la fila) para recuperar el valor original (equivale a precargar la fila para el siguiente acceso a memoria).



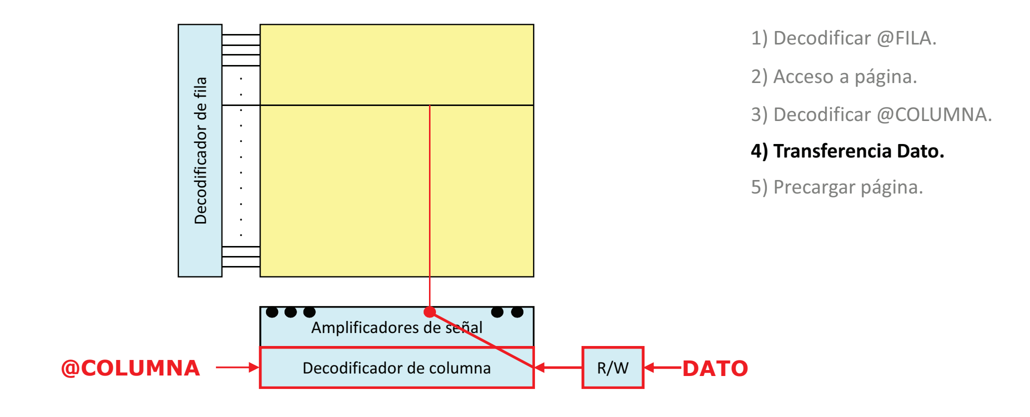
1) Decodificar @FILA.
2) Acceso a página.
3) Decodificar @COLUMNA.
4) Transferencia Dato.
5) Precargar página.



Estructura interna de una DRAM

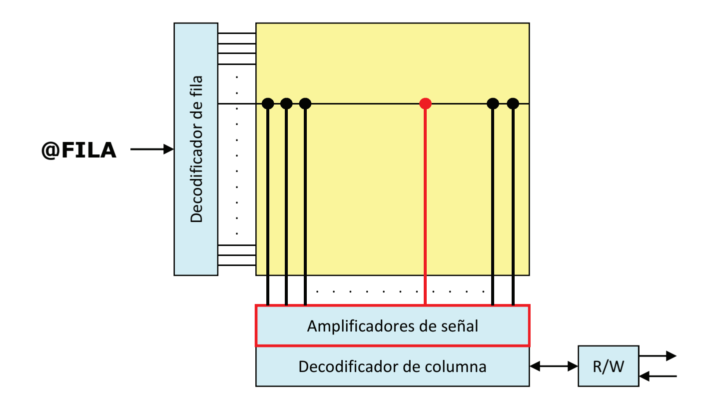
- **Una operación de escritura:**

  3 y 4) Prácticamente igual, la única diferencia es que la celda se reescribe con el dato que entra por el buffer R/W.



Estructura interna de una DRAM

- **Una operación de escritura:**
  5) Hay que reescribir la celda con el nuevo valor (y el resto de la fila con el valor original).

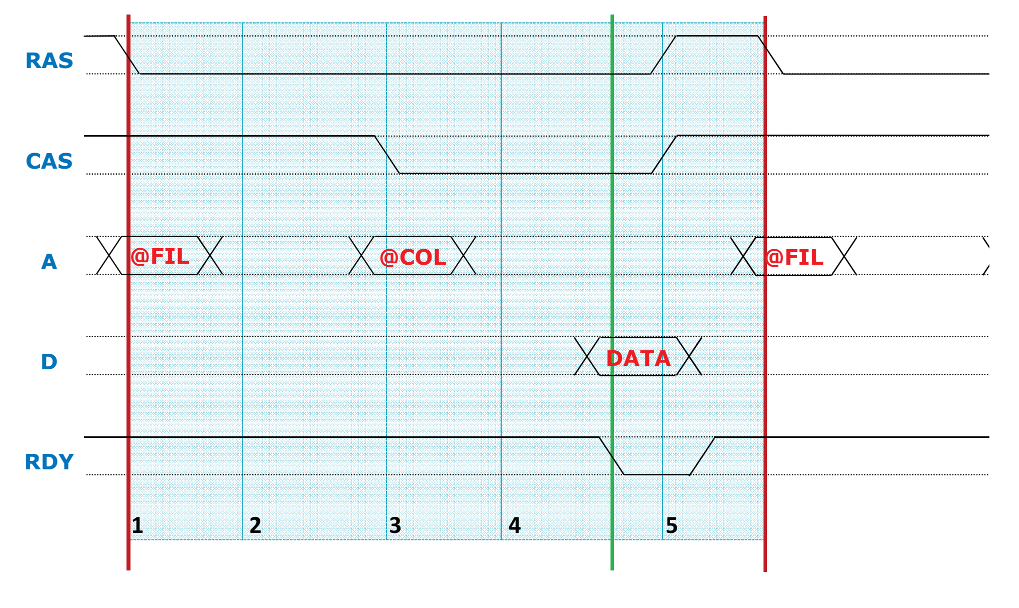


1) Decodificar @FILA.
2) Acceso a página.
3) Decodificar @COLUMNA.
4) Transferencia Dato.
5) Escribir/Precargar página.



Estructura interna de una DRAM

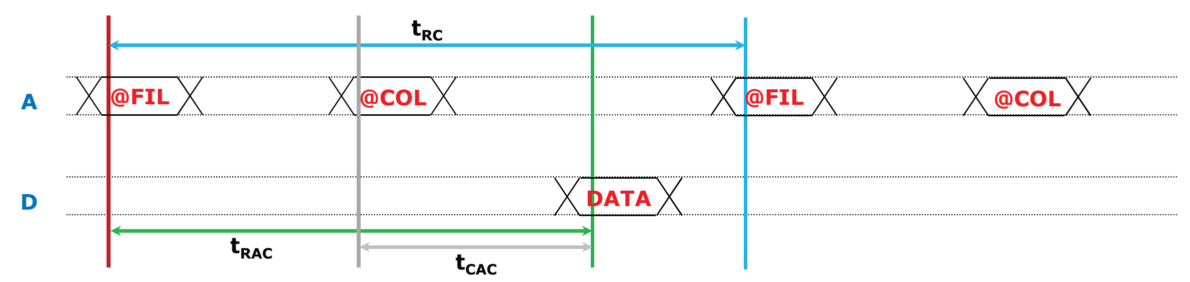
Cronograma simplificado de una operación de lectura



1) Decodificar @FILA.
2) Acceso a página.
3) Decodificar @COLUMNA.
4) Transferencia Dato.
5) Precargar página.

Estructura interna de una DRAM

Cronograma simplificado de una operación de lectura



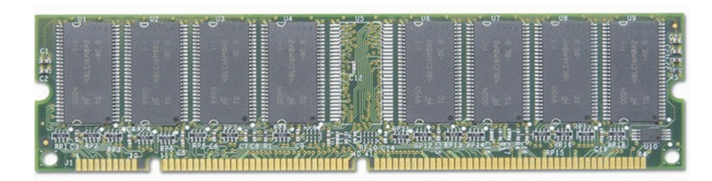
3 Valores fundamentales:

- **Tiempo de acceso** (t_{RAC}): retardo máximo desde que se suministra la dirección de fila hasta que se obtiene el dato → **latencia de memoria**.
- **Tiempo de ciclo** (t_{RC}): intervalo de tiempo mínimo entre dos accesos consecutivos a memoria → **ancho de banda**.
- **Tiempo de acceso a columna** (t_{CAC}): retardo máximo desde que se suministra la dirección de columna hasta que se obtiene el dato.



Evaluación y optimización

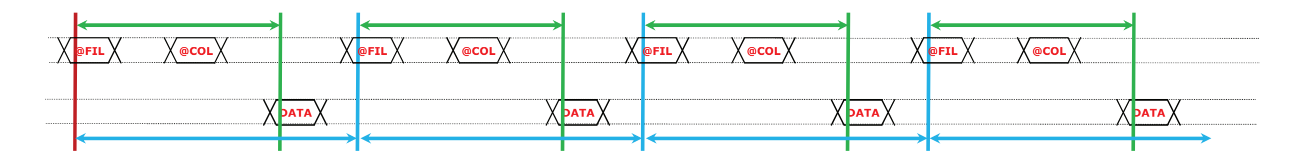
- **La memoria principal suele estar organizada en DIMMs**
- **Se accede en paralelo a todos los chips . En un acceso se accede (p.e.) a 64 bits (8B)**



- **¡Todos los accesos a memoria principal son para leer/escribir líneas de cache!**
- **Tamaño típico de una línea de cache: 32B**
- **Si queremos leer una línea de cache:**
  - **¡Es necesario realizar 4 accesos consecutivos a memoria!**

Evaluación y optimización

- **Leer una línea de cache: 4 lecturas de memoria principal**
  - Tiempo de acceso: 50ns
  - Tiempo de ciclo: 60ns



- **Coste de leer una línea de cache:**
  - Tiempo total: Tiempo de ciclo · 3 + Tiempo de acceso: 230 ns
  - Ancho de banda: 32B / 230ns = 139.13 MB/s

- **Posible optimización: los datos que leemos de componen una línea de cache están en posiciones consecutivas de memoria**

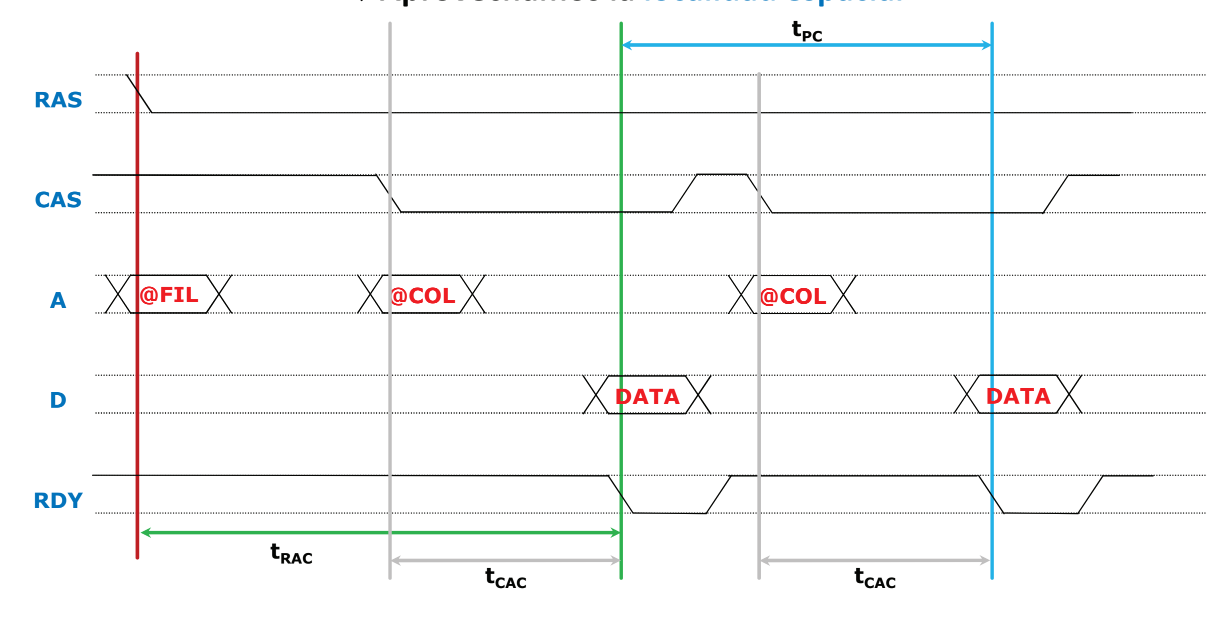
  → **Están en la misma fila del array de memoria (en posiciones consecutivas)**



Evaluación y optimización

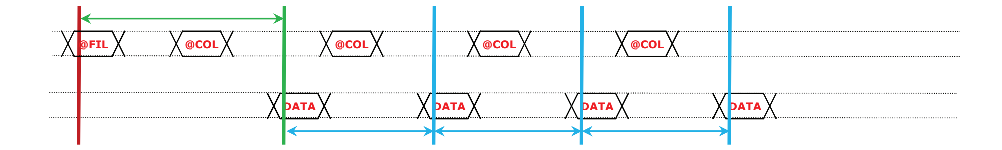
- **Idea Fundamental: una vez accedida la fila, se puede acceder a varias columnas simplemente cambiando la @COL.**

  → **Aprovechamos la localidad espacial**



Evaluación y optimización

- **Leer una línea de cache: 4 lecturas de memoria principal**
  - Tiempo de acceso: 50ns
  - Latencia de columna: 30ns



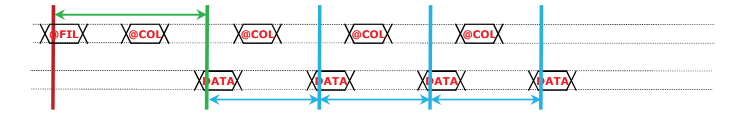
- **Coste de leer una línea de cache:**
  - Tiempo total: Tiempo de acceso + Latencia de columna · 3 : 140 ns
  - Ancho de banda: 32B / 140ns = 228.57 MB/s
  - Ancho de banda de pico: 8B / 30ns = 266.67 MB/s

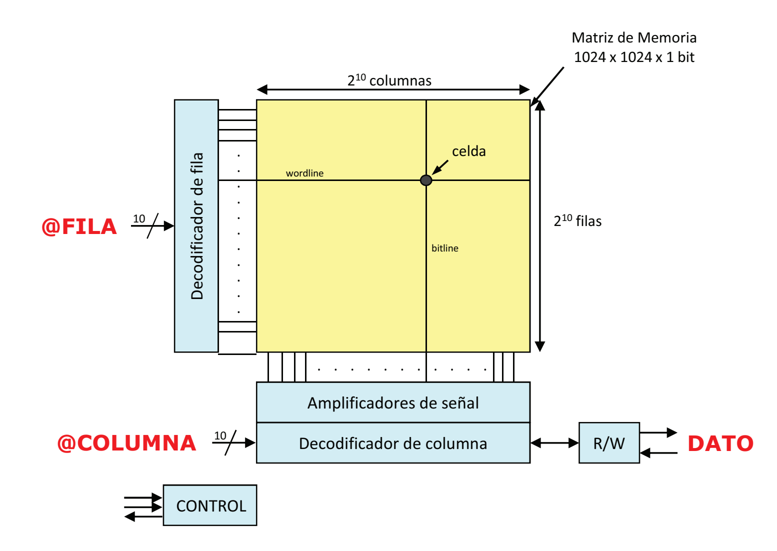
- **La latencia del primer dato se mantiene.**
- **Mejora la latencia de los siguientes datos simplemente cambiando el protocolo de acceso a memoria. El hardware es exactamente el mismo.**



Evaluación y optimización

- **Problema : hay que esperar a que el dato sea leído antes de enviar la nueva @COL.**

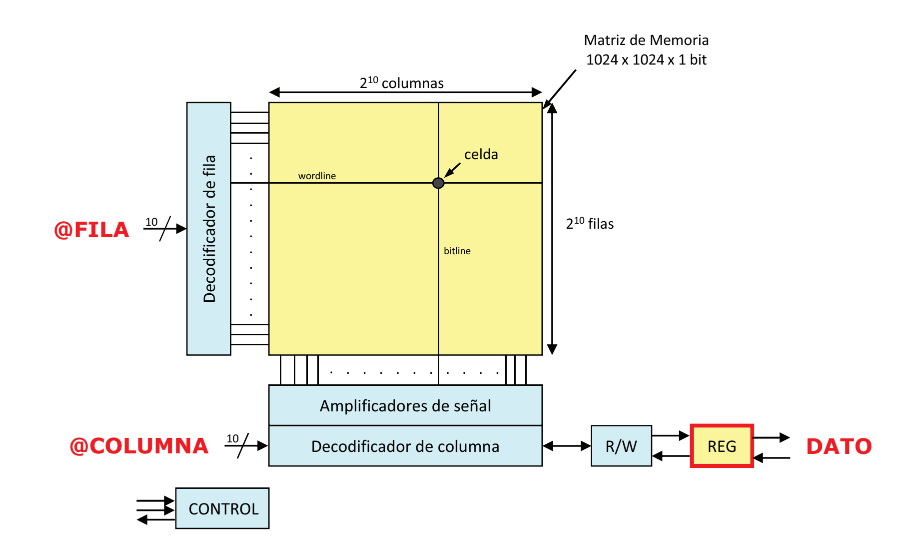




Evaluación y optimización

- **Solución**: se añade un registro en la salida de datos

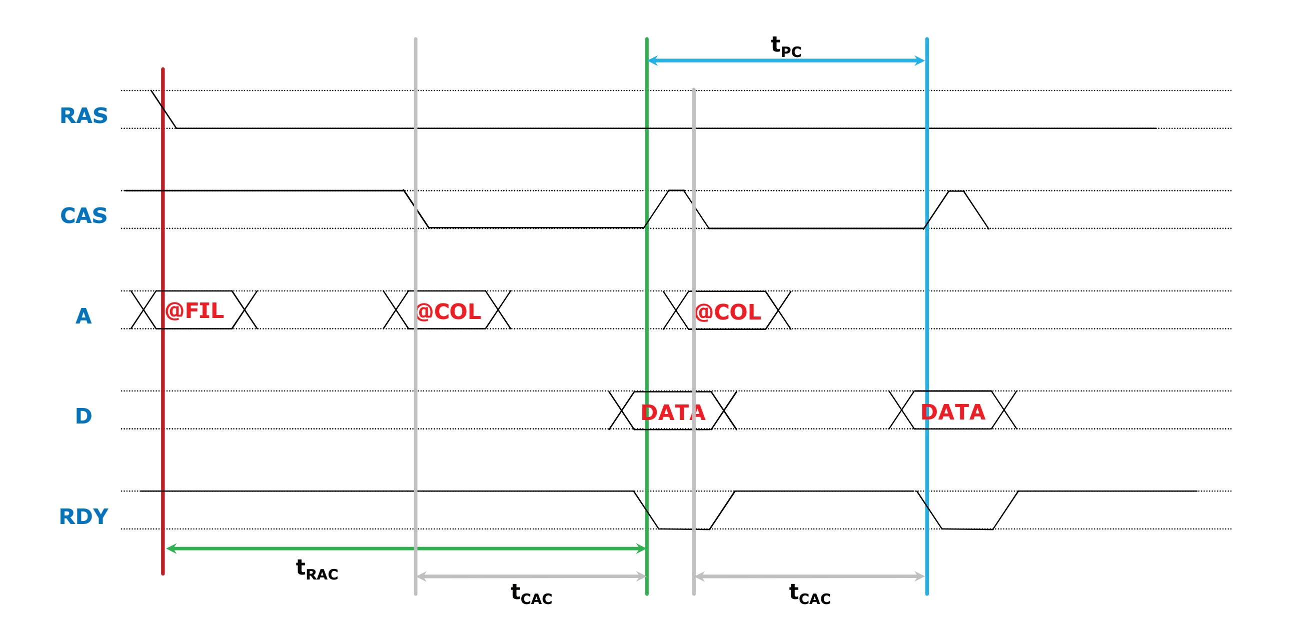
  → se puede solapar el acceso a los datos con el envío de la nueva @COL





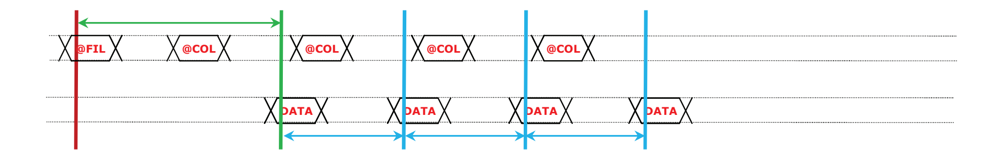
Evaluación y optimización

- **Idea Fundamental**: Como el dato está almacenado en un registro, antes de acabar el envío del dato ya podemos enviar la @COL.



Evaluación y optimización

- **Leer una línea de cache: 4 lecturas de memoria principal**
  - Tiempo de acceso: 50ns
  - Latencia de columna: 20ns



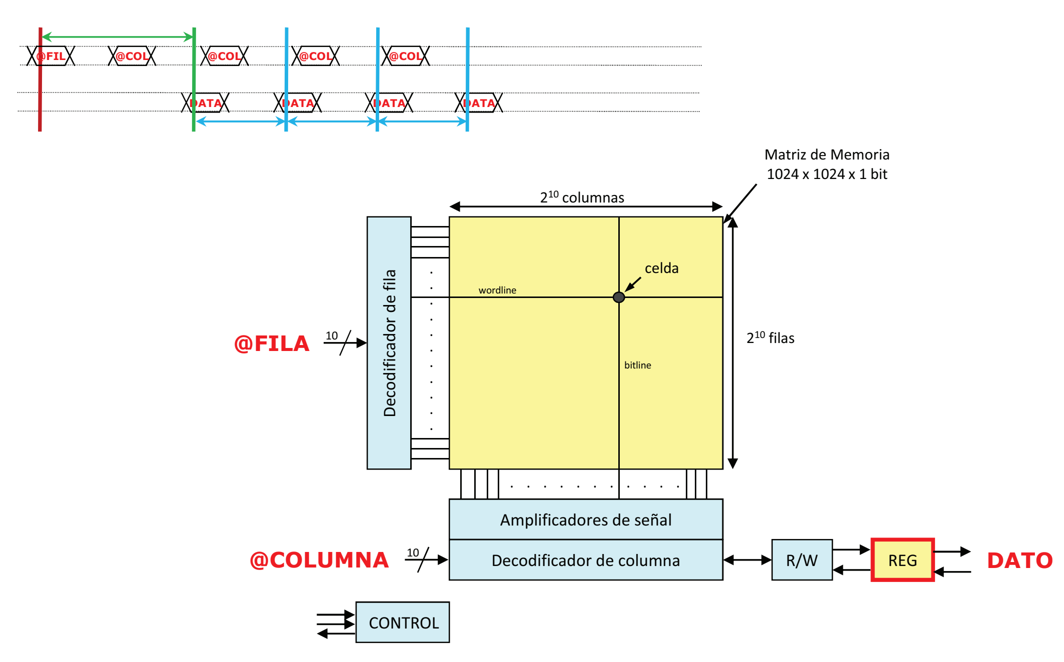
- **Coste de leer una línea de cache:**
  - Tiempo total: Tiempo de acceso + Latencia de columna · 3 : 110 ns
  - Ancho de banda: 32B / 110ns = 290.91 MB/s
  - Ancho de banda de pico: 8B / 20ns = 400 MB/s

- **La latencia del primer dato se mantiene.**
- **Mejora la latencia de los siguientes datos simplemente añadiendo un registro.**



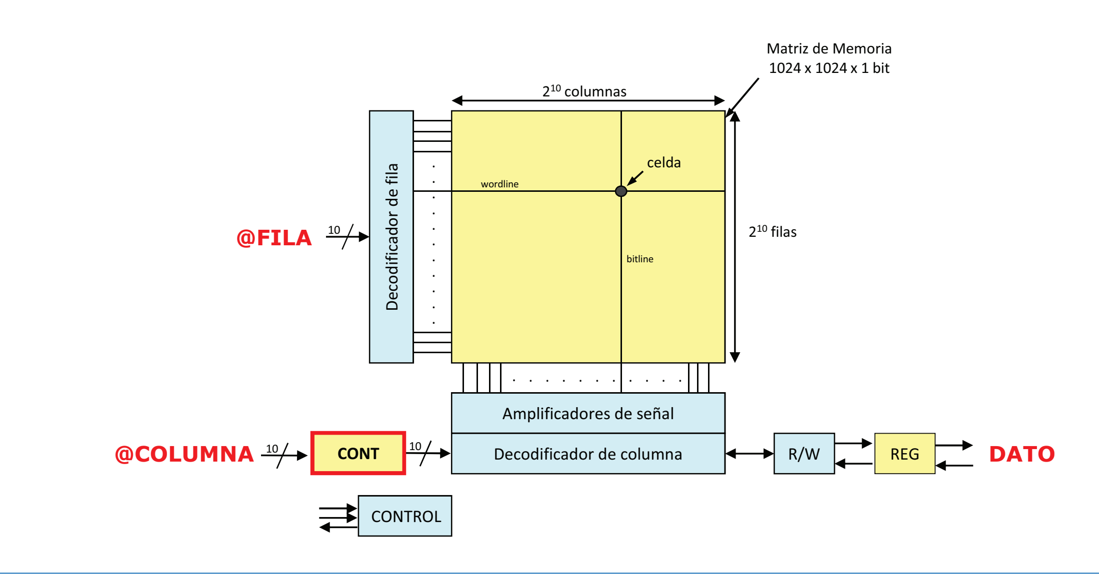
Evaluación y optimización

- **Problema: Hay que enviar la @COL para cada dato. Pero cuando se accede a 1 línea de cache estamos accediendo a @COL, @COL+1, @COL+2 y @COL+3.**



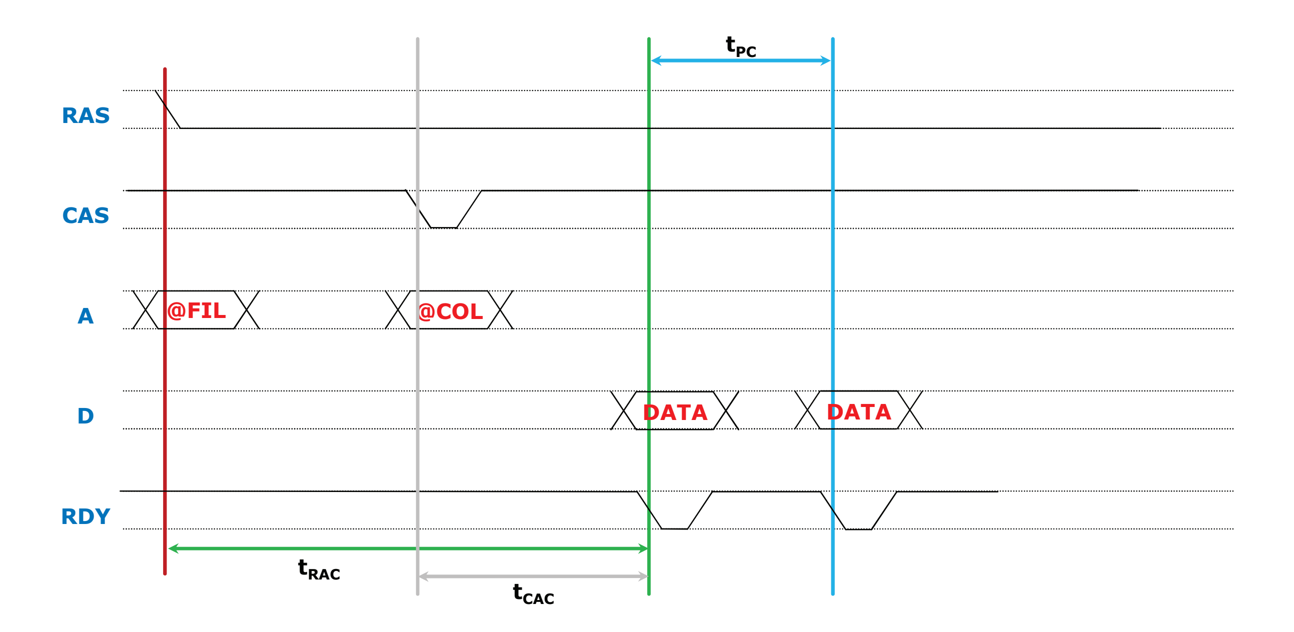
Evaluación y optimización

- **Solución: Añadir un contador para que genere de forma automática @COL+1, @COL+2 y @COL+3.**



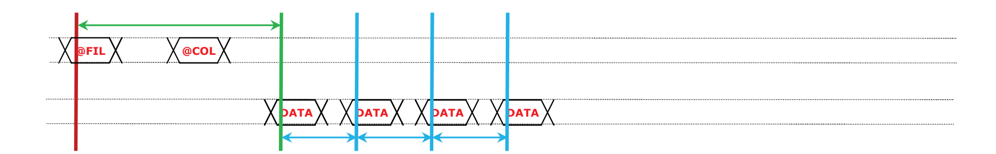
Evaluación y optimización

- **Idea Fundamental: Sólo hay que enviar la @COL una vez, del resto se encarga la propia memoria.**



Evaluación y optimización

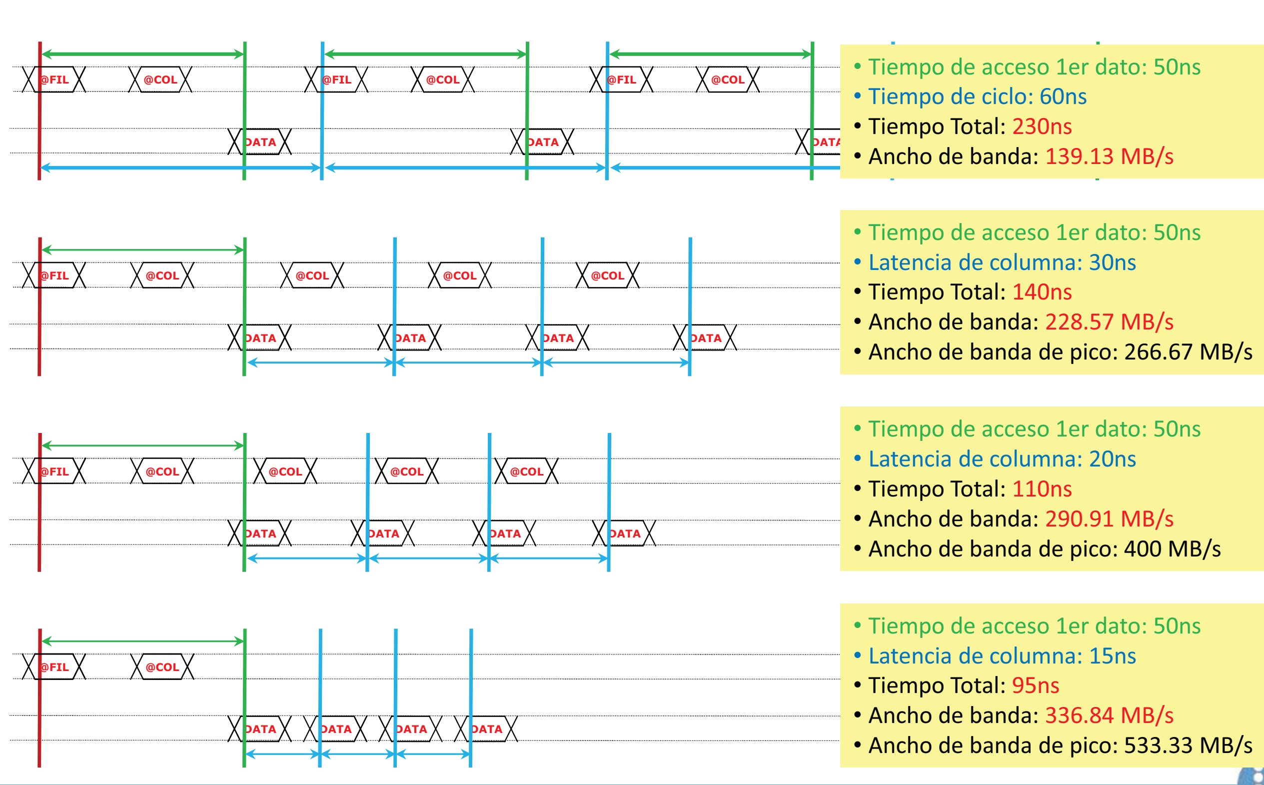
- **Leer una línea de cache: 4 lecturas de memoria principal**
  - Tiempo de acceso: 50ns
  - Latencia de columna: 15ns



- **Coste de leer una línea de cache:**
  - Tiempo total: Tiempo de acceso + Latencia de columna · 3 : 95 ns
  - Ancho de banda: 32B / 95ns = 336.84 MB/s
  - Ancho de banda de pico: 8B / 15ns = 533.33 MB/s

- **La latencia del primer dato se mantiene.**
- **Mejora la latencia de los siguientes datos simplemente añadiendo un registro.**
- **En este punto se llega al límite de rendimiento de una memoria ASÍNCRONA.**

Evaluación y optimización



Evaluación y optimización

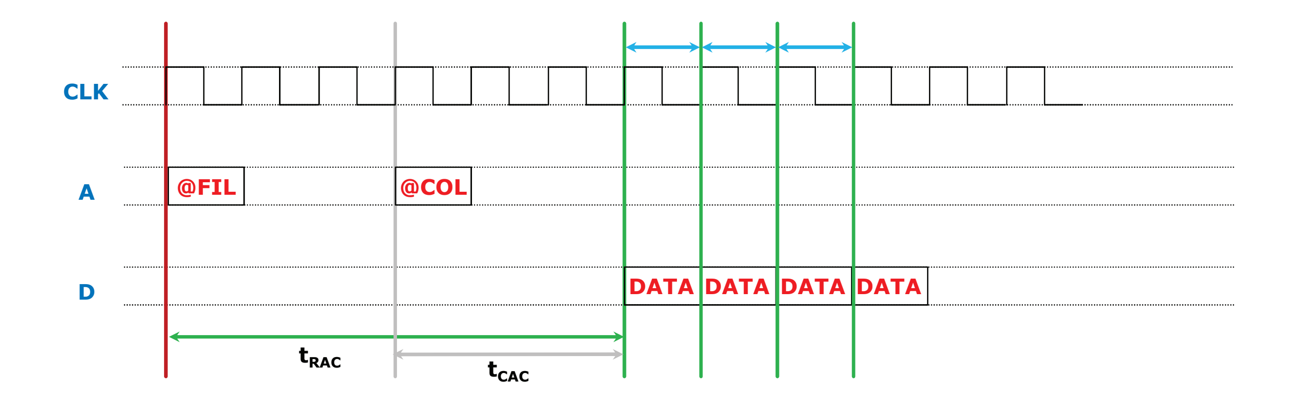
Problema de las memorias asíncronas:

- El protocolo de comunicación con la memoria es costoso (lento).
- Cada paso (de sincronización) requiere:
  - ✓ Enviar el parámetro (p.e. @FIL)
  - ✓ Esperar que se estabilice
  - ✓ Enviar la señal de sincronismo (p.e. RAS)
  - ✓ Obtener el parámetro
- Hay 4 pasos de sincronización (FIL, COL, DATA, PRE)
- Los tiempos (de acceso, ciclo, etc) son muy difíciles de reducir por problemas de ruido.
- Las memorias asíncronas más veloces se usaban en placas base a 66 MHz

Evaluación y optimización

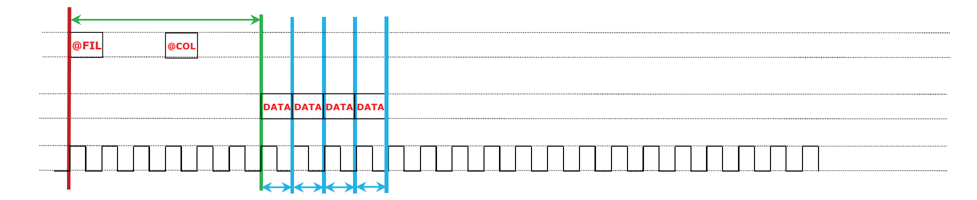
Solución: hacer que la memoria funcione de forma SÍNCRONA

- El protocolo se simplifica, desaparecen los problemas de ruido
- Se puede aumentar la frecuencia de funcionamiento



Evaluación y optimización

- **Leer una línea de cache: 4 lecturas de memoria principal**
  - Tiempo de acceso: 40ns
  - Latencia de columna: 10ns

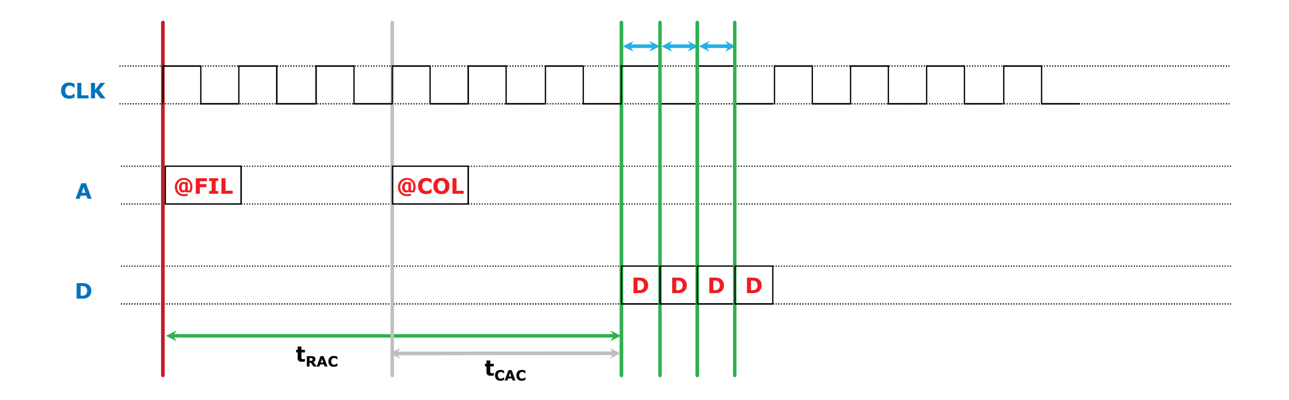


- **Coste de leer una línea de cache:**
  - Tiempo total: Tiempo de acceso + Latencia de columna · 3 = 70ns
  - Ancho de banda: 32B / 70ns = 457.14 MB/s
  - Ancho de banda de pico: 8B / 10ns = 800 MB/s

- **La latencia del primer dato mejora un poco.**
- **La latencia de los siguientes datos mejora de forma sustancial. Además, el margen de mejora es muy grande.**

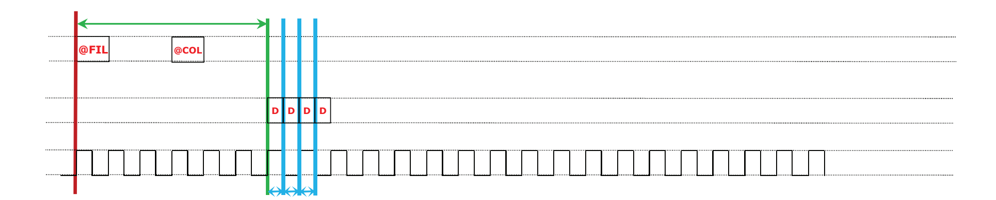
Evaluación y optimización

- **Mejora: modificando la salida de datos se puede aumentar de forma sustancial el ancho de banda obtenido.**



Evaluación y optimización

- **Leer una línea de cache: 4 lecturas de memoria principal**
  - Tiempo de acceso: 40ns
  - Latencia de columna: 5ns



- **Coste de leer una línea de cache:**
  - Tiempo total: Tiempo de acceso + Latencia de columna · 3 = 55ns
  - Ancho de banda: 32B / 55ns = 581.81 MB/s
  - Ancho de banda de pico: 8B / 5ns = 1600 MB/s

- **La latencia del primer dato es la misma.**
- **Un pequeño cambio en el hardware de la salida de datos permite doblar el ancho de banda de pico de la memoria.**

FPM DRAM
- Ancho de banda: 228.57 MB/s
- Ancho de banda de pico: 266.67 MB/s

EDO DRAM
- Ancho de banda: 290.91 MB/s
- Ancho de banda de pico: 400 MB/s

BEDO DRAM
- Ancho de banda: 336.84 MB/s
- Ancho de banda de pico: 533.33 MB/s

SDRAM
- Ancho de banda: 457.14 MB/s
- Ancho de banda de pico: 800 MB/s

DDR SDRAM
- Ancho de banda: 581.841 MB/s
- Ancho de banda de pico: 1600 MB/s

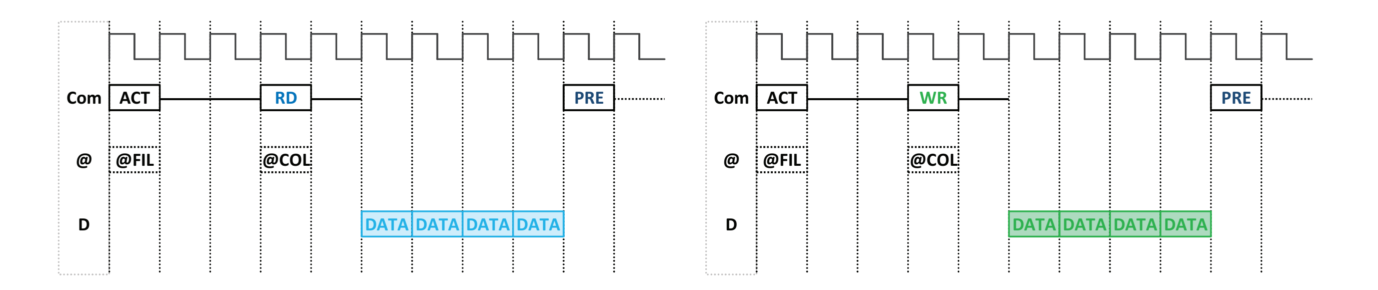
- El array de memoria es prácticamente el mismo. La latencia del primer dato se mantiene.
- La gran mejora se obtiene en el acceso a los siguientes datos.

RETOS:
- Reducir la latencia del primer dato
- ¿Cómo aprovechar el ancho de banda de pico de una DDR?

Funcionamiento estándar de una SDRAM

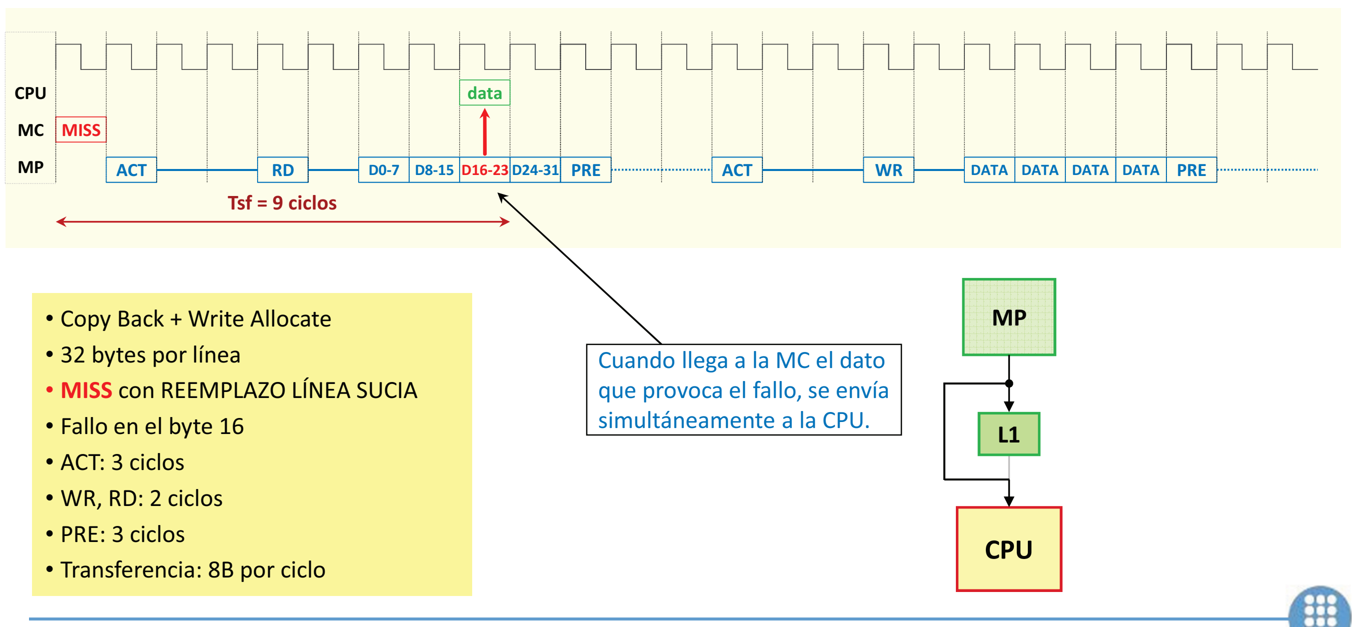
- Todas las SDRAM, incluyendo los diferentes tipos de DDR funcionan con comandos:
  - ACTIVE (ACT)
  - READ (RD)
  - WRITE (WR)
  - PRECHARGE (PRE)

- Cronogramas de acceso en Lectura y en Escritura



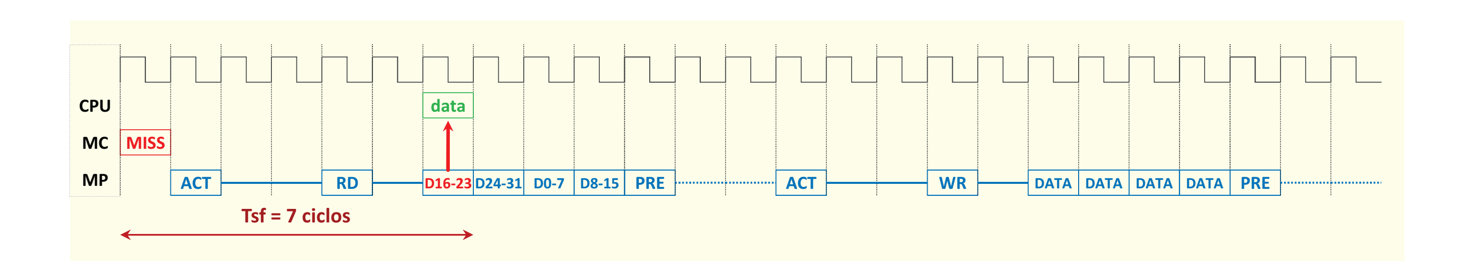
Revisitando Cronogramas

- **Continuación Anticipada** (*Early Restart*): en cuanto llega el dato que provoca el fallo, se envía al procesador



Revisitando Cronogramas

- **Transferencia en desorden: se envía en primer lugar la palabra que ha provocado el fallo.**



- Copy Back + Write Allocate
- 32 bytes por línea
- MISS con REEMPLAZO LÍNEA SUCIA
- Fallo en el byte 16
- ACT: 3 ciclos
- WR, RD: 2 ciclos
- PRE: 3 ciclos
- Transferencia: 8B por ciclo

En todos los casos, y para que estos mecanismos sean efectivos, cuando se pasa el dato al procesador y mientras se acaba de servir el fallo, la MC ha de ser capaz de recibir nuevos accesos (Cache no bloqueante).

